Universidade Federal do Rio de Janeiro

Instituto Tércio Pacitti de Aplicações e

Pesquisas Computacionais

Alexandre Luis Ribeiro Chagas

Rede Veicular: VANET Hibrida, 3G e 4G e Geoposicionamento a Serviço da Segurança Pessoal

Rio de Janeiro

2016

Alexandre Luis Ribeiro Chagas

Rede Veicular:

VANET Hibrida, 3G e 4G e Geoposicionamento a Serviço da Segurança Pessoal

Monografia apresentada para obtenção do título de Especialista em Gerência de Redes de Computadores no Curso de Pós-Graduação Lato Sensu em Gerência de Redes de Computadores e Tecnologia Internet do Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro – NCE/UFRJ

Orientador:

Claudio Miceli de Farias, DSc, UFRJ, Brasil

Rio de Janeiro

2016

Alexandre Luis Ribeiro Chagas

Rede Veicular:

VANET Hibrida, 3G e 4G e Geoposicionamento a Serviço da Segurança Pessoal

Monografia apresentada para obtenção do título de Especialista em Gerência de Redes de Computadores no Curso de Pós-Graduação Lato Sensu em Gerência de Redes de Computadores e Tecnologia Internet do Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais da Universidade Federal do Rio de Janeiro – NCE/UFRJ

Aprovada em março de 2016

Claudio Miceli de Farias, DSc, UFRJ, Brasil

Dedico este projeto a humanidade em busca de um futuro pacifico e tecnologicamente eficaz para proteger o ser de si mesmo.

AGRADECIMENTOS

Agradeço a minha esposa por toda a força e apoio físico, moral e sentimental dados a mim especialmente dentro dos dois anos de duração dessa especialização. Também por sua paciência pelo confinamento em casa em nome de estudos, por servir de escudo contra vários percalços durante a mesma duração e que poderiam, sem sua intervenção, ter provocado a interrupção deste grande projeto.

Agradeço ao Prof. Moacyr, Gerente Executivo do curso e responsável por sua administração, todo o cuidado e auxilio dados nas matérias, sua preocupação com o transcurso e desempenho e principalmente pelo nível de excelência do profissionalismo apresentado.

Agradeço ao meu orientador Prof. Claudio Miceli, pelo nível de suas aulas, pelo apoio, pela orientação, pela coragem, e principalmente por acreditar neste trabalho.

Por fim, mas mais importante, Agradeço a Deus, que me proporcionou todas as condições necessárias para a conclusão do presente curso de especialização, que me protegeu e que me deu forças durante todo o período.

RESUMO

Chagas, Alexandre Luis Ribeiro **REDE VEICULAR: VANET Hibrida, 3G e Geoposicionamento a Serviço da Segurança.** Monografia (Especialização em Gerência de Redes e Tecnologia Internet). Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais, Universidade Federal do Rio de Janeiro Rio de Janeiro, 2016.

O presente projeto visa o estudo no uso das redes veiculares hibridas compostas pelas VANETS infraestruturada e *adhoc,* assim como as redes de dados móveis 3 e 4G's com o objetivo de suas utilizações no auxílio da segurança do motorista e do patrimônio em colisões, capotamentos, saída de estradas em lugares ermos e possivelmente sequestros, ou seja, qualquer situação de ameaça ou risco ao já citado.

ABSTRACT

Chagas, Alexandre Luis Ribeiro **VEHICULAR NETWORK: VANET Hybrid, 3G and geopositioning** of the security service Monograph (specialization in Network Management and Internet Technology) Institute Tércio Pacitti Applications and Research Computing, Federal University of Rio de Janeiro Rio de Janeiro, in 2016.

This project aims to study the use of hybrid vehicle networks composed by VANETS infra-structured and ad hoc as well as mobile data networks 3 and 4G's with the purpose of their use in aid of driver safety and property in collisions in rollovers, road out in the wilderness and possibly kidnapping, or any threat or risk to the already mentioned.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3G	Terceira Geração
4G	Quarta Geração
ABS	Antilock Braking System
AMP	Arbitration on Message Priority
AP	Access Point
CAN	Controller Area Network
CDMA	Code Division Multiple Access
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
DSRC	Dedicated Short Range Communications
ENIAC	Eletronic Numerical Integrator And Computer
ECU	Engine Control Unit
FQ	Fair Queuing
IEEE/ I3E	Institute of Eletrical and Eletronics Engineers
IVG	Inter Vehicles Geocast
GPS	Global Positioning System
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HSPA	High Speed Packet Access
MIMO	Multiple-Input Multiple-Output
PARC	Palo Alto Reserch Center
QoS	Quality of Service
RSCP	Rate-Controlled Static Priority
RR	Round Robin
RSU	Road Side Unit
VANET	Vehicular Ad hoc Network
WAVE	Wireless Access in the Vehicular Enviroments
WFQ	Weighted Fair Queuing
WiFi	Wireless Fidelity
WRR	Weighted Round Robin

SUMÁRIO

1 INTRODUÇÃO

"O futuro dependerá daquilo que fazemos no presente!" - Mahatma Gandhi.

A evolução da humanidade ocorre cada vez mais rápido. Desde a Era Industrial demos grandes saltos em pesquisas nas mais variadas áreas, mas, nenhuma delas evoluiu tanto e tão rápido quanto a área tecnológica, afinal, deve-se considerar que suas ferramentas foram responsáveis pela evolução de todas as outras áreas.

Dentre todas, três evoluções nos interessam no presente trabalho, a Computacional, das Redes e a Automotiva Paralelo a evolução Computacional e principalmente, provocada por ela, houve a evolução das redes de computadores. Sua criação adveio da necessidade de transporte rápido, confiável e em maior quantidade de dados de um computador para outro. Antes de sua criação essa transferência era feita através do uso de cartões perfurados que armazenavam poucos caracteres, o que complicava o transporte das informações devido a quantidade de cartões a serem transportados.

Em 1970 foi criada por Normam Abramson na universidade do Havai a ALOHANet que interligava as ilhas do Havaí através de micro-ondas por rádio, ou seja, a primeira rede não dependente de cabos ou fios, totalmente *wireless.* Com protocolos de controle próprios foi a primeira rede externa a receber um IMP (*interface message processors*) e integrar a ARPANet[16]. O que começou com micro-ondas, tornou-se hoje uma gama de padrões complexos que foram desenvolvidos ao longo dos anos pelo IEEE/I3E, no grupo de trabalho 80211 e que é responsável pelos padrões 11a, 11b, 11g, 11i, 11n e atualmente 11ac, além de outros que popularizaram as redes sem fio e sua aplicação em várias tecnologias[5].

A informatização da indústria automotiva foi a responsável pelos inúmeros avanços presentes na evolução dos veículos atuais. A evolução dessa tecnologia não se limitou apenas aos motores mas, ao conforto proporcionado aos ocupantes. Passou-se de bancos de madeira a luxuosos estofamentos de couro, do som da rua a equipamentos sonoros de alta fidelidade e pureza, e como principal, de painéis com mostradores simples à computadores de bordo integrados com GPS (*Global Positioning System)* orientados a voz, sensores de estacionamento, conexão de telefonia por Bluetooth, gerenciamento de desempenho e informação de defeitos.

1.1 MOTIVAÇÃO

Como devidamente demonstrado, a convergência de todas as tecnologias citadas são consequência direta da evolução de cada uma, logo, devemos direcionar esta convergência em prol do bem pessoal, qual seja, a segurança do indivíduo em primeiro lugar.

Já em 1991, Mark Weiser, cientista chefe e diretor de tecnologia da empresa Xerox PARC[1](*Palo Alto Reserch Center*), apresentou ao mundo o conceito da "computação UBIQUA"[2] que é o cerne da convergência da evolução computacional ao dia-a-dia do homem, mas, de forma transparente ao usuário, não dependente de controle ou intermediação para execução da função e alcance do resultado [20].

Este conceito nos apresenta - já naquele ano -, que o homem, suas vestes, seus equipamentos, veículos, sinais de trânsito, placas indicativas, afinal tudo que o rodeava tornar-se-ia um nó endereçável tanto de uma rede privada como da internet, o limite seria dado apenas pelo alcance de dada rede.

Sua concretização ocorreu mais rapidamente na evolução automobilística que integrou a seus projetos cada vez mais sensores, o que tornou obrigatório o

[1] Divisão independente da Xerox Corp.

[2] Computação Ubíqua é a integração da informática a vida, ações e comportamentos naturais do homem. Cf. https://www.ics.uci.edu/~corps/phaseii/Weiser-Computer21stCentury-SciAm.pdf

desenvolvimento de uma rede de controle para que os dados enviados não sofressem colisões entre mensagens prioritárias como sensores de frenagem, aceleração, controle de velocidade e serviços que afetam o motor.

A rede CAN (C*ontroller Area Network)*, criada por Robert Bosh em1980, foi aprimorada tornando-se padrão internacional em 1994 através da ISO 11898, para controlar este tráfego a nível de aplicação[1]. Todo o acima nos demonstra que embora ainda não executada, podemos utilizar as VANETS em qualquer de suas configurações, assim como a Rede de telefonia celular 3G e a Rede de satélites de geoposicionamento para trafegar certos dados que visem a segurança do motorista no aviso de possíveis perigos no caminho, acidentes por tombamento e capotamento, colisões, e até situações de roubo, furto e sequestro.

1.2 OBJETIVO

Uma vez que são parcos os estudos da utilização de várias tecnologias de comunicação simultâneas no campo automotivo, o presente trabalho tem como objetivo apresentar sua viabilidade sem o detrimento de qualquer uma delas em favor de outra, ou ainda, sem interferência entre elas quando da utilização.

Para tal, analisaremos os protocolos de transmissão, de segurança uma vez que nem todos os dados serão encaminhados e não se quer dar acesso à rede CAN do veículo a controle externo. A possibilidade de envio de mensagem de emergência sobre essas tecnologias e a confecção de um App para externalização do mesmo sinal.

1.3 METODOLOGIA

Seguir-se-á com o estudo por explanação das tecnologias existentes na rede veicular e seus protocolos, na rede de controle automotivo, e finalmente a construção e aplicação de APP para atendimento ao objetivo a ser alcançado.

1.4 ORGANIZAÇÃO

Nos capítulos a seguir, trataremos a revisão bibliográfica com o intuito de mostrar a base de motivação e o ponto de partida do presente trabalho, conduzindo o leitor a através das VANETS Hibridas e seus protocolos.

Trataremos os conceitos do intercâmbio entre as tecnologias, a segurança necessária para evitar invasões no sistema, a construção do aplicativo de controle e finalmente a conclusão do trabalho apresentando a viabilidade do estudo.

2 REVISÃO BIBLIOGRAFICA

No presente capítulo, passamos a analisar os estudos e publicações correntes referentes a matéria discutida.

2.1 REDE AUTOMOTIVA

A rede automotiva é responsável pelo tráfego e controle dos dados gerados por módulos veiculares, responsáveis pelo funcionamento dos equipamentos segurança, conforto, diagnóstico e afins, cada mais utilizados nos veículos modernos. Dividem-se em classes sendo elas A, B e C.

A rede Classe A é mais utilizada pelas funções de conforto e diagnóstico Em Santos 2010 essas redes são definidas da seguinte forma:

"São redes de comunicação com baixa largura de banda utilizadas em funções de conforto e diagnóstico Geralmente para essas funções a tendência é ter como acessar uma das redes padrão do veículo como vidros elétricos, retrovisores, controle de bancos, lâmpadas, etc. Alguns exemplos de redes automotivas Classe A são UART, BEAN, ABUS, LIN, TTP/A entre outras" (Santos 2010, p72).

Tabela 1: Tecnologias de Redes Automotivas Classe A
Fonte: Santos, 2010

Empresa	UART (ALDL)	SINEBUS	I^2C	SAE J1708	CCD	ACP	BEAN	LIN	TTP/A	A-BUS
	GM	DELCO	PHILIPS (NXP)	TMC-ATA	Chrysler	FORD	TOYOTA	Consórcio	TTTech	VW
Aplicação	Geral Diagnóstico	Áudio	Controle Diagnóstico	Controle Diagnóstico	Geral Diagnóstico	Áudio Controle	Carroceria Controle Diagnóstico	Carroceria Controle Diagnóstico	Carroceria Controle Diagnóstico	Carroceria Diagnóstico
Meio Físico	Simples Fio	Simples Fio	Par Trançado	Par Trançado	Simples Fio	Par Trançado	Simples Fio	Simples Fio	Simples Fio	-
Código Bit	NRZ	SAM	AM	NRZ	NRZ	NRZ	NRZ	NRZ	-	-
Controle Acesso Meio	Mestre / Escravo	Mestre / Escravo	Mestre / Escravo	Mestre / Escravo	Mestre / Escravo	Mestre / Escravo	Contenção	Mestre / Escravo	Mestre / Escravo	-
Controle Erro	8-bit CS	Nenhum	bit ACK	8-bit CS	8-bit CS	8-bit CS	8-bit CRC	8-bit CS	-	-
Cabeçalho	16 Bits	2 Bits	-	16 Bits	8 Bits	12-24 Bits	25 Bits	2 Bits/Byte	16 Bits	-
Dados	0-85 Bytes	10-18 Bits	-	-	5 Bytes	6-12 Bytes	1-11 Bytes	8 Bytes	16 Bytes	-
Overhead	Variável	75%	45%	Variável	16.7%	25%	28%	2 Bytes	39%	-
Taxa Transmissão	8192 bps	66.6kHz - 200kHz	1-100 kbps	9600 bps	7812.5 bps	9600 bps	10 kbps	20 kbps	50 kbps	-
Max. Compr. Barramento	Não Especificado	10 m	Não Especificado	Não Especificado	Não Especificado	40 m	Não Especificado	40 m	-	-
Max. Nº ECU	10	-	-	-	6	20	20	16	-	-

A rede classe B é normalmente utilizada em funções relacionadas à dinâmica do veículo. São definidas ainda em Santos 2010. São redes utilizadas para aplicações importantes para a operação do automóvel e não demandam elevados requisitos de comunicação de dados. Geralmente são utilizadas para interconectar ECU's (*Engine Control Unit*) que gerenciam unidades como motor, transmissão, *retarder*, embreagem, etc. Alguns exemplos de rede automotivas classe B são VAN, J1850, J1939 E CAN.

Tabela 2: Tecnologias de Redes Automotivas Classe B
Fonte: Santos, 2010

Empresa	SINGLE-WIRE CAN (SWC)	CAN 2.0 ISO 11898 ISO 11519-2 ISO 11992 J2284	J1850 ISO 11519-4			SAE J1939	VAN
	SAE/ISO	Bosch/SAE/ISO	GM	FORD	Chrysler	TMC-ATA	PSA/Renault
Aplicação	Diagnóstico	Controle Diagnóstico	Geral Diagnóstico	Geral Diagnóstico	Geral Diagnóstico	Diagnóstico	Controle Diagnóstico
Meio Físico	Simples Fio	Par Trançado	Simples Fio	Par Trançado	Simples Fio	Simples Fio	Par Trançado
Código Bit	NRZ-5 MSB first	NRZ-5 MSB first	VPW MSB first	PWM MSB first	VPW MSB first	NRZ-5 MSB first	NRZ-5 MSB first
Controle Acesso Meio	Contenção	Contenção	Contenção	Contenção	Contenção	Contenção	-
Controle Erro	CRC	CRC	CRC	CRC	CRC	CRC	-
Cabeçalho	11 Bits	11 ou 29 Bits	32 Bits	32 Bits	8 Bits	11 Bits	-
Dados	0-8 Bytes	0-8 Bytes	0-8 Bytes	0-8 Bytes	0-10 Bytes	0-8 Bytes	-
Overhead	9.9%	9.9% - 22%	33.3%	33.3%	8.3%	9.9%	-
Taxa Transmissão	33.3 kbps 83.33 kbps	10 kbps to 1 Mbps	10.4 kbps	41.6 kbps	10.4 kbps	33.33 kbps 83.33 kbps	-
Max. Compr. Barramento	30 m	Não Especificado 40 m (típico)	35 m (5m para scan tool)	35 m (5m para scan tool)	35 m (5m para scan tool)	30 m	-
Max. Nº ECU	16	Não Especificado 32 (típico)	32	32	32	16	-

A rede classe C é normalmente utilizada para funções de segurança crítica. A definição das redes é dada por Santos 2010 como:

“Redes utilizadas em aplicações de segurança crítica com requisitos de tempo real e tolerância a falhas que estejam diretamente ligadas à dinâmica do automóvel e à segurança ativa. Aplicações baseadas na tecnologia *x-by-wire* requerem que as redes ofereçam transmissão de dados com baixo atraso, alta frequência, tolerância a faltas e outros mecanismos essenciais para segurança crítica” (Santos 2010, P 75).

Tabela 3: Tecnologias de Redes Automotivas Classe C
Fonte: Santos, 2010

Empresa	TTP/C	FlexRay	TT-CAN	ByteFlight	BST	Safe-by-Wire	DSI
	TTTech	Consórcio	CiA	BMW	Siemens Bosch/Temic	Delphi/Philips TRW/Autoliv SDI	Motorola
Aplicação	Airbag	Segurança Crítica	Segurança Crítica	Segurança Crítica	Segurança Crítica	Segurança Crítica	Segurança Crítica
Meio Físico	Par Trançado	Par Trançado Fibra Óptica	Par Trançado	Fibra Óptica	Dois Fios	Dois Fios	Dois Fios
Código Bit	-	NRZ	NRZ	-	Manchester Bifásico	3 Níveis Tensão	3 Níveis Tensão
Controle Acesso Meio	TDMA	F-TDMA	TDMA	F-TDMA	-	-	-
Controle Erro	CRC 16 Bits	CRC 24 Bits	CRC 15 Bits	CRC 16 Bits	Paridade CRC	CRC 8 Bits	CRC 4 Bits
Dados	16 Bytes	0-246 Bytes	0-8 Bytes	-	1 Byte	1 Byte	1-2 Bytes
Overhead	-	-	32	-	-	64	16
Taxa Transmissão	5-25 Mbps	10 Mbps	1-2 Mbps	10 Mbps	31.25 kbps 125 kbps ou 250 kbps	150 kbps	5 kbps 150 kbps
Max. Compr. Barramento	-	25 m	40 m	-	-	25-40 m	-
Max. Nº ECU	64/256	-	32	-	12 squibs + 62 slaves	64	16

2.1.1 Barramento CAN

Para a devida compreensão do estudo a seguir, é necessária a revisão sobre o sistema de controle da rede veicular interna para que se possa definir quais as informações seguras para transmissão sem comprometimento da segurança do veículo e de seus ocupantes.

A rede CAN é responsável pela administração dos sinais gerados diretamente pelos sensores ou pelos módulos que compõe um veículo.

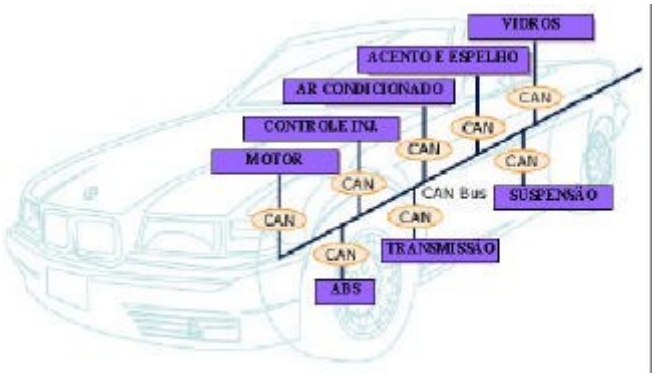

Figura 1 - Rede CAN em um automóvel [8]

A figura acima mostra que o veículo é composto por vários módulos interligados e controlados pela rede CAN. Esses módulos são as ECU's responsáveis pela administração da rede de dados do veículo. A figura 11 apresenta arquitetura distribuída que é a interligação de várias ECU's sendo cada qual responsável pela administração de função especifica do veículo. Serão abordadas mais à frente.

2.2 PROTOCOLOS AUTOMOTIVOS

A Rede CAN é capaz de utilizar variados protocolos de gerenciamento que mudam de acordo com cada fabricante. Alguns desses protocolos utilizam o padrão OSI de sete camadas previsto na ISO 7498-1:1994:

Figura 2 - padrão OSI de sete camadas ISO 7498-1/94 [3]

Os protocolos de comunicação são definidos por GUIMARÃES (2011):

"Protocolos de comunicação são meios de transmissão e recepção de dados utilizados para intercomunicar módulos eletrônicos e/ou sensores e atuadores inteligentes equipados com micro controladores e transceivers por exemplo Existem vários tipos de protocolos de comunicação cada qual com suas características técnicas especificas e, portanto, com suas aplicações mais apropriadas"

2.2.1 Protocolo CAN

Protocolo serial síncrono que utiliza detecção de colisão CSMA/CD (*Carrier Sense Multiple Access with Collision Detection)* + NDA (*Non-Destructive Arbitration)* no barramento CAN. O controle é feito através da leitura do meio antes de cada transmissão o que só ocorre após a constatação de meio livre, mas se houver transmissões simultâneas, o módulo com menor prioridade cessa sua transmissão permitindo que o de maior prioridade continue enviando mensagem sem a necessidade de reiniciá-la.

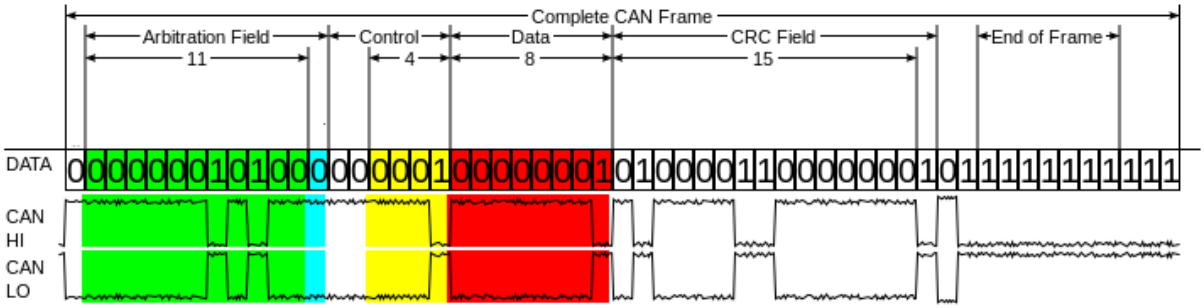

Figura 3 - campos de identificação do pacote do protocolo CAN [4]

A velocidade de transmissão de dados no barramento é inversamente proporcional ao comprimento do mesmo A maior taxa de transmissão especificada no CAN é de 1 mega byte considerando-se um barramento de tamanho máximo de 40 metros.

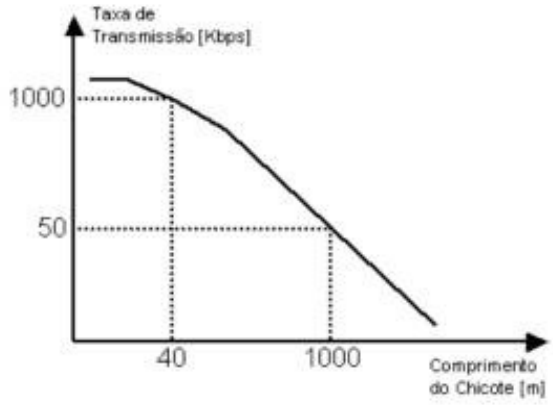

Figura 4 - inversão de proporção de velocidade por comprimento [8]

2.3 REDES MÓVEIS

Embora o presente trabalho busque a utilização de rede móvel celular, esta não será objeto de estudo direto em seus modos, meios, protocolos de controle ou difusão. As redes moveis se caracterizam por ter seus nós móveis e por se conectarem por meio de rádio. Dividem-se em dois tipos: As Adhoc e as infraestruturadas.

As redes Adhoc se caracterizam por não necessitarem de infraestrutura já que seus nós se interconectam diretamente uns com os outros. Por seus nós serem móveis, seu protocolo de controle deve ser altamente adaptável quanto as rotas de encaminhamento devido as substituições dos nós que saíram do alcance da rede por outros que entraram. Já, as infraestruturadas, dependem de um AP que centraliza e controla o tráfego da rede por protocolos de roteamento e por controle de acesso ao meio.

2.4 REDE VEICULAR

As redes veiculares são compostas por nós móveis controlados por sistemas embarcados[3] ou dispositivos adaptados com esta capacidade. Existem atualmente dois tipos de rede veicular: VANET (Veicular Ad hoc Network) pura e a VANET Híbrida.

Para entender o conceito das VANETS, precisamos primeiro entender o que é uma MANET (Mobile Ad Hoc Network). Uma MANET tem essa nomenclatura porque trabalha exclusivamente com enlaces Adhoc, que como explicado tem interconexão direta entre os nós que se auto organizam, mas, sua mobilidade é restrita fazendo com que os nós se mantenham conectados de forma estável por mais tempo.

[3] Sistema com objetivo definido encapsulado em outro sistema computacional. Origem na computação ubiqua. Wieber, *op. cit.* pag. 15.

As VANET's são a extensão do uso das MANET's para o sistema veicular com algumas alterações de protocolo considerando que os nós, por serem motorizados, movimentam-se a grandes velocidades A VANET pura utiliza apenas os nós Adhoc enquanto a Hibrida utiliza tanto os nós móveis quanto a infraestrutura disponibilizada ao longo de todo o trajeto percorrido.

No modelo puro a conectividade depende diretamente da densidade de trafego de veículos que utilizam essa tecnologia, já que o encaminhamento ocorre através de múltiplos saltos, como no caso em que um destino está fora do alcance da origem, este só será alcançado através dos nós intermediários que receberam e retransmitirão os pacotes, o que é desvantajoso em comparação ao modelo hibrido.

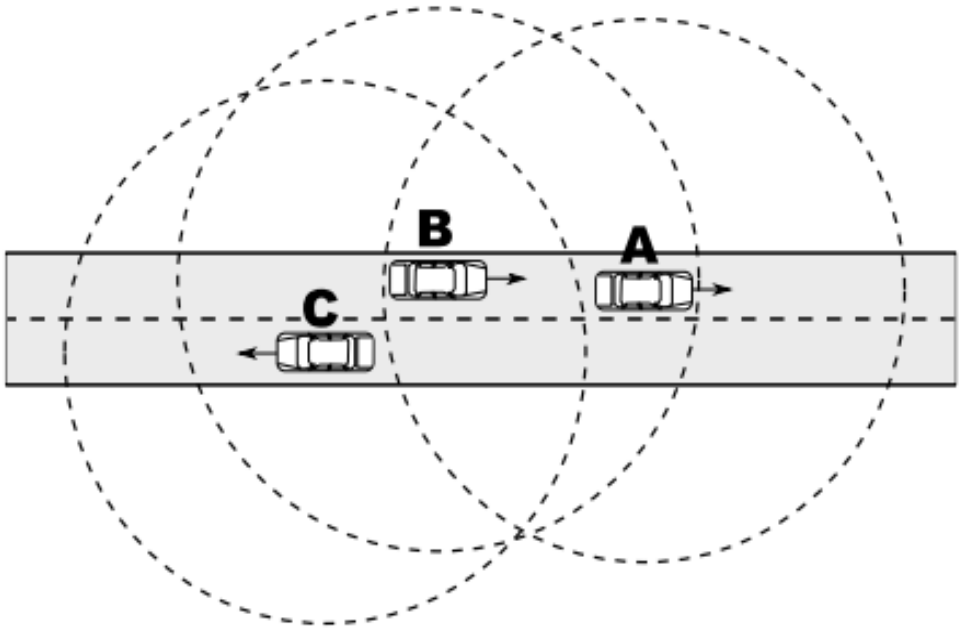

Figura 5 - alcance de transmissão entre os nós móveis (CORREIA; CELESTINO; CHERKAOUI, 2011)

Já, no modelo infraestruturado, a utilização dos RSU (*Road Side Unit*) supre a necessidade de alta densidade de veículos, já que ela será a responsável por encaminhar o dado recebido a outra RSU a qual esteja conectada por cabo ou por

outro meio sem fio ou ainda a outro veículo ao seu alcance como demonstrado na figura a seguir:

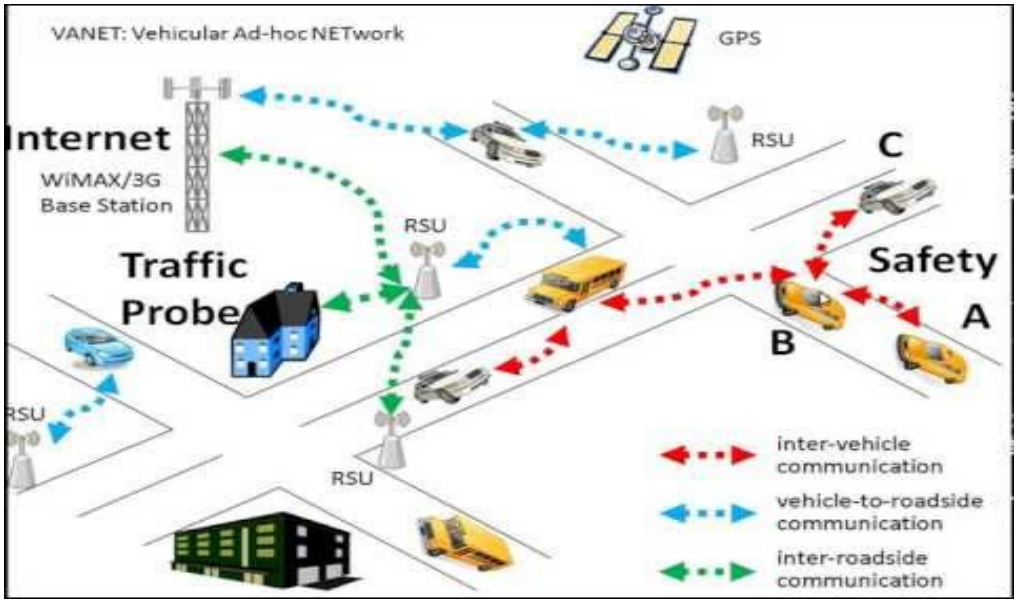

Figura 6 - Uso da VANET Hibrida no cotidiano [11]

2.4.1 Padrões

As redes veiculares apresentam vários padrões de acordo com a região e o país, alguns dos quais são apresentados abaixo:

2.4.2 Dedicated Short Range Communications - DSRC

Espectro de banda cuja o objetivo precípuo era de fornecer comunicações de curto alcance Estruturado em 7 canais de 10MHz (UZCATEGUI; ACOSTA-MARUM, 2009).

Esses canais são:

- 172 emergência e preservação da vida
- 174/176/180/182 Canais de serviço
- 178 Canal de controle
- 184 alta potência e segurança publica

A comutação entre eles é definida pela arquitetura wave e feita por divisão de frequência e tempo enviados em períodos repetitivos de 100 ms. Como se verifica na figura abaixo, a cada 100 ms, 50 ms são alocados para o canal de controle (CCH) e outros 50 ms são alocados para os canais de serviços (SCH), incluindo 4 ms de intervalo de guarda para a comutação entre os canais CCH e SCHs (LI, 2010).

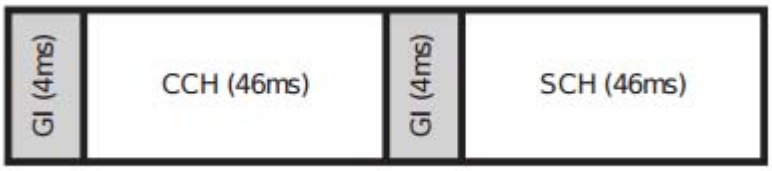

Figura 7 - comutação de canais DSRC definida pela arquitetura wave

2.4.3 Wireless Access in the Vehicular Enviroments - WAVE

A arquitetura WAVE padronizada em 2004 pelo I3E, é definida nos seguintes estudos: 16091, 16092, 16093, 16094 e 80211p O padrão 80211p define as camadas físicas e de controle de acesso ao meio (MAC) para redes veiculares. Mas a arquitetura WAVE não se restringe às camadas MAC e física. Os padrões da família IEEE 1609 definem outras camadas da pilha de protocolos, incluindo uma camada de rede alternativa à camada IP, características a segurança para aplicações DSRC e operação em múltiplos canais de comunicação (UZCATEGUI; ACOSTA-MARUM, 2009; ALVES et al, 2009).

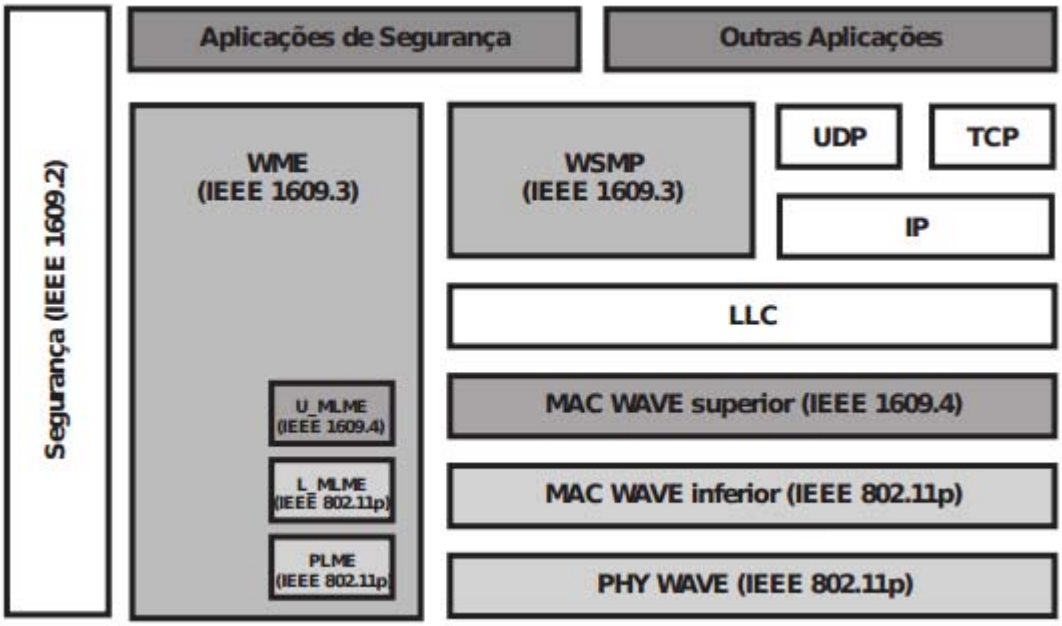

Figura 8 - Arquitetura Wave com aplicações de segurança

3 ROTEAMENTO

Devido a constante troca dos nós em uma VANET, o roteamento da informação transmitida é um verdadeiro teste para todos os protocolos que controlam os enlaces físicos, tanto para as mensagens curtas (de segurança), quanto para download de dados, ou seja, maior quantidade de tráfego, uma vez que a quantidade, direção e a velocidade de deslocamento dos nós são altamente variáveis.

O modo de transmissão dos dados em uma VANET pode ocorrer de quatro formas: no modo Unicast, que é a conexão direta entre os nós origem/destino, sem a utilização de hops e com baixo overhead; no modo broadcast, utilizado pelas mensagens de segurança que precisam ser enviadas para todos os nós ao alcance; no modo multicast, onde o envio se dá entre um grupo pré-cadastrado; e no geocast, a mensagem é enviada para todos os nós que se encontrem em uma determinada posição geográfica, relativa ao nó que envia a mensagem, e este último modo merece nossa atenção no presente trabalho. Dentre todos os protocolos utilizados em geocast, conforme a literatura, o IVG é o que melhor se apresenta para o fim a que se destina:

3.1 INTER-VEHICLES GEOCAST - IVG

O IVG (BACHIR; BENSLIMANE, 2003) é usado para informar a todos os veículos que estão em uma estrada sobre qualquer perigo que ocorra, principalmente um acidente A área de risco é definida com base na direção, posicionamento e velocidade dos veículos. Veículos localizados na área de risco formam um grupo multicast.

O protocolo IVG usa broadcasts periódicos para superar possíveis fragmentações da rede e entregar a mensagem para todos os membros do grupo

criado naquele evento O protocolo IVG reduz a quantidade de hops, fazendo com que os nós mais distantes tenham prioridade no acesso ao meio para a entrega mais rápida ao grupo A Figura abaixo mostra um exemplo de como funciona o protocolo IVG.

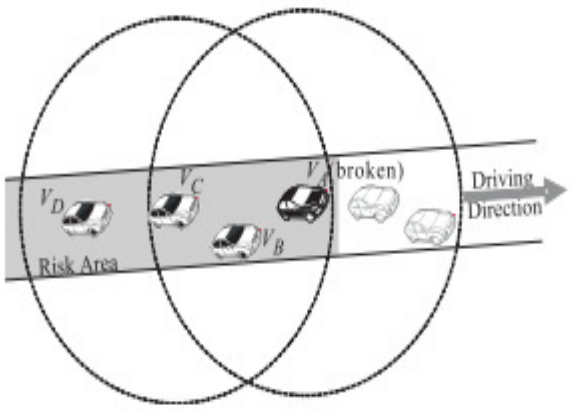

Figura 9 - Dinâmica do protocolo IVG [19]

O veículo VA encontra um problema e envia para todos os veículos na área de risco. Os veículos VB, VC e VD formam o grupo porque estão na área de risco. O responsável por encaminhar as mensagens de VA é o VC porque ele está mais longe do que o VB. O veículo VB recebe as mensagens de VA, mas não as encaminha. Isso reduz a quantidade de saltos na transmissão de mensagens para a área de risco.

3.2 DO COMPARTILHAMENTO DE RECURSOS

Na VANET o compartilhamento de recursos nos modos de transmissão é muito exigido em vista da mudança dos nós. Isso duplica as ocorrências de contenção, o que obriga a utilização de QoS para priorizar o desempenho de aplicações de segurança e em caso de não utilização, por outros de menor prioridade.

Para tal, os algoritmos de escalonamento são essenciais uma vez que gerenciam as filas de serviço e organizam a ordem de execução na competição,

desempenhando um papel importante nos diferentes níveis de QoS para as aplicações do meio, equilibrando o tráfego não prioritário e dando a devida ênfase ao prioritário quando necessário.

3.3 DOS ALGORITMOS DE ESCALONAMENTO

Os algoritmos de Escalonamento estão divididos em *Work-conserving Non-work-conserving.* A diferença entre eles está relacionada com a forma como se definem os períodos de inatividade, ou seja, se é considerado que o nó está inativo apenas quando não existe nada para transmitir (*work-conserving*), ou se o nó pode estar inativo mesmo que existam pacotes a serem transmitidos (*non-work-conserving*) [11].

No escalonamento Work-conserving devemos citar:

- *Fair Queueing (FQ)*, caracteriza-se por marcar os pacotes na chegada, com a estimativa do instante em que o pacote deve voltar a sair. Desta forma os fluxos serão ordenados pelas marcas efetuadas à chegada;

- *Weighted Fair Queueing (WFQ)*, é um algoritmo idêntico ao FQ, a diferença é que o WFQ permite ter diferentes partilhas de serviços para diferentes sessões;

- *Round-Robin (RR)*, que se caracteriza por servir rotativamente um pacote de cada fila não vazia. É um protocolo justo se os pacotes tiverem comprimento fixo e se as ligações tiverem o mesmo peso, imune ao problema de *starvation*;

- *Weighted Round-Robin (WRR)*, que se caracteriza por servir as ligações proporcionalmente aos pesos atribuídos, e pressupõe o conhecimento do tamanho médio dos pacotes gerados. É justo apenas em escalas temporais superiores à duração do ciclo, podendo ser injusto durante ciclos longos para ligações com um peso pequeno.

No escalonamento *Non-work-conserving* devemos citar:

- *Stop-and-Go* [15] e [19], define de vários níveis para alocar recursos em um único nó, tentando garantir a fluidez do tráfego em toda a rede.

- *Rate-Controlled Static Priority (RCSP)* [15] e [19], é um algoritmo mais complexo que o *Stop-and-Go.* Este algoritmo separa as funções em dois componentes: (1). Tem um regulador que controla o tráfego de distorção introduzido pelos efeitos de multiplexação e variações de carga em outros nós e 2). É um regulador de prioridade para multiplexar o tráfego regulado.

A utilização do algoritmo de escalonamento garante que as mensagens de acidente tenham prioridade na rede.

4 GEOLOCALIZAÇÃO

Como as VANET's são compostas por nós móveis, é necessário que se saiba constantemente a localização por controle da posição de cada nó constante da rede, um exemplo desse controle pode ser obtido através do uso do GPS.

Com os disparos dos *beacons*, a posição de cada nó é informada a seus vizinhos tornando possível o encontro do destino pelo nó transmissor sem a utilização de endereçamento fixo como o IP, o que torna desnecessário o estabelecimento de rotas para todos os nós da rede.

Mas em verdade, existe uma facilitação para esse controle uma vez que já foi feito o mapeamento de estradas - pelo menos nas principais -, sabemos que pegando certa estrada em certa localidade, ter-se-á direção de deslocamento e possíveis direções a serem tomadas, portanto, variáveis que auxiliam a tomada de decisão de para onde o envio do pacote deve ser direcionado.

5 DA PROPOSTA

Mesmo com todas as inovações e avanços nas tecnologias acima especificadas, ainda hoje temos desaparecimentos e mortes em colisões veiculares fora do perímetro urbano que por falta de comunicação provoca a demora no socorro ocasionando falecimento dos ocupantes do veículo.

Mas a presente solução apresenta dois problemas: (i) como externar o aviso ou a mensagem de perigo ou de emergência originada pelos Sensores do veículo automotor utilizando a rede CAN previamente instalada para as VANETS exibidas e para os serviços 3 ou 4G; (ii) como prover a segurança do sistema veicular uma vez que a rede CAN contém todas as ECU's responsáveis por funções críticas do veículo, quais sejam, motor, dirigibilidade, frenagem, sinalização ou iluminação.

Considerando ser a emissão de um sinal de emergência de alta prioridade, este deverá ser somente de saída dando maior Liberdade e Menor controle. É real também o perigo de invasão do mesmo sistema veicular, onde através de invasão do próprio sistema de emergência poderia o invasor ter acesso às ECU's, tendo aí a capacidade de provocar acidentes atentando contra a vida dos que estão dentro do veículo; ocasionar paradas irregulares para assaltos e sequestros através do corte do motor; interromper sinalização e iluminação entre outros.

Como exemplo, foi apresentado por Charlie Miller e Chris Valacek, Engenheiro de segurança do *Twitter* e diretor de pesquisas de cibersegurança da *IOActive*, respectivamente, em experimento fechado, a invasão do sistema *Uconnect* desenvolvido pelo grupo Chrysler.

Utilizando um *smartphone*, invadiram a Rede CAN veicular através do computador de bordo e interromperam o funcionamento dos freios e da transmissão de um *Jeep Cherokee*[4].

Para solucionar esses problemas, não podemos permitir que o sistema de emergência utilize a rede CAN veicular, mas podemos partilhar alguns dos sensores nela existentes para constituição de uma rede de emergência com ECU especifica que servirá de modulo de leitura veicular e emissor das mensagens de emergência tanto para a VANET, quanto para os dispositivos móveis reconhecidos por ela.

Não será desenvolvido no presente trabalho a ECU acima referenciada, mas podemos dizer que terá duas formas de contato com o mundo externo: por *wi-fi* e por *bluetooth.* A fonte de energia do modulo será a própria bateria do veículo automotor, mas, conterá uma pequena bateria de backup que entrará em funcionamento caso no acidente ou colisão a bateria principal do veículo seja danificada.

O modulo enviará simultaneamente o sinal de emergência tanto para o dispositivo móvel pessoal cadastrado, quanto para as VANETS ao alcance e continuará enviando enquanto o sistema estiver sendo alimentado por energia.

Pelo acima exposto é fácil concluir que, se o presente sistema não passa ou não interfere com a rede CAN, é impossível a um pretenso invasor interferir em qualquer sistema do veículo, mesmo que consiga invadir a rede "CRASH", o que em caso positivo, fará com que o mesmo só interfira sobre ela, habilitando-a ou desabilitando-a.

Para a solução do primeiro problema, passo a apresentar o app para o dispositivo móvel antes referenciado.

[4] Para mais detalhes, visite: http://oglobo.globo.com/economia/tecnologia/hackers-conseguem-invadir-sistema-de-carros-conectados-controla-los-remotamente-16872780

5.1 DO APP DE EMERGÊNCIA

Deverá ter uma interface simples de utilização, visualização e identificação, mas sem a intenção de chamar a atenção, a tela escura é proposital para não iluminar ambientes escurecidos indicando posicionamento do usuário.

Inicialmente será nomeado de "*CrashLocator*" contendo apenas dois botões em sua interface. Um botão maior no meio nomeado de "Emergência" e um segundo botão nomeado de "Configurar" na parte inferior direita da tela do dispositivo.

Figura 10 - interface primaria do app

Esse segundo botão abre uma nova tela apresentando 3 campos de inserção de número de telefone para mensagem SMS, 3 para e-mails e 3 para contatos do aplicativo *WhatsApp* para os quais serão enviadas todas as mensagens de emergência.

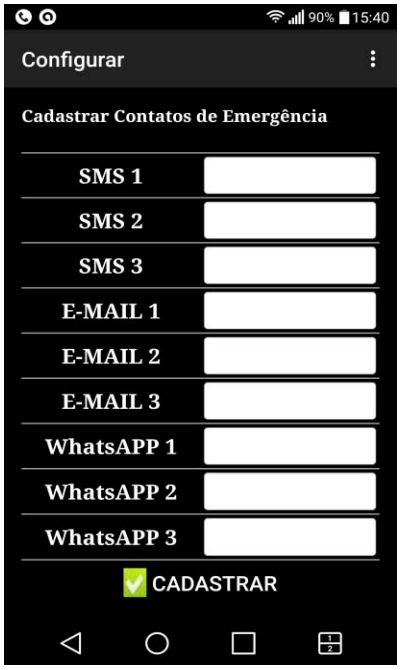

Figura 11 - tela de configuração do app

O app no dispositivo móvel deverá ficar habilitado em segundo plano todo tempo.

Sua conexão com o dispositivo veicular, como já dito, será através de *bluetooth* e *wi-fi* e funcionará de duas formas:

Habilitado em segundo plano ele fará contato com telefone ou e-mail pré-cadastrado de forma automática assim que receber a mensagem de alerta do módulo veicular. Poderá também em caso de emergência ser acionado diretamente do dispositivo móvel através do botão emergência, enviando uma segunda mensagem para números previamente cadastrados

Em ambos os casos as mensagens enviadas pelo dispositivo móvel aos números pré-cadastrados conterão a triangulação do posicionamento no momento do envio da mensagem, ou seja, conterão as coordenadas do momento do acionamento seja pelo modo veicular seja pelo acionamento do botão de emergência do App.

Figura 12 – Mensagem de aviso de Emergência para os contatos cadastrados

No caso de acionamento pelo módulo veicular a mensagem enviada pelo dispositivo móvel será a mesma e sofrerá start nas seguintes situações:

- Colisão frontal, colisão traseira, colisão lateral e capotamento

- A colisão frontal será identificada pelo acelerômetro instalado na ECU da Rede “Crash”;

- As colisões traseiras e laterais serão identificados também pela instalação de acelerômetro no modulo e por um sensor colocado no para-choque traseiro do veículo;

- Capotamentos serão identificados através na colocação de 1 sensor de giro angular no módulo veicular que identificará a modificação na posição de 180 graus do ponto inicial acionando a mensagem de capotamento.

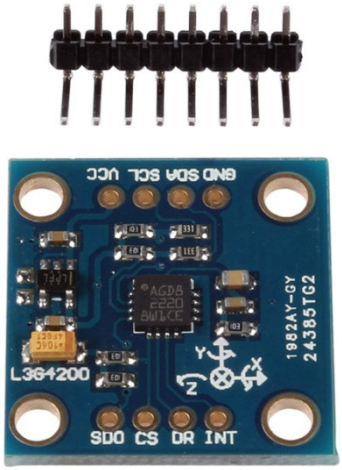

Figura 13 - sensor de giro angular sobre os eixos x, y e z

6 CONCLUSÃO

Como restou devidamente demonstrado a união da evolução da tecnologia automotiva e das redes de comunicação, nos apresenta condições de melhorar a segurança dos condutores de veículos utilizando soluções privadas imediatas, sem a necessidade de solicitação de autorizações governamentais para utilização de meios oficiais.

Foi comprovado por experimentação que o objetivo do presente estudo foi atingido, uma vez que o aplicativo desenvolvido apresentou funcionamento satisfatório com margem de erro mínimo quanto ao resultado de geoposicionamento, tanto quando o veículo permaneceu em campo aberto, como quando foi colocado em espaços restritos e cobertos como garagens particulares e de acesso público.

7 DOS TRABALHOS FUTUROS

No presente trabalho, apresentamos o desenvolvimento e a execução do aplicativo responsável pelo envio das mensagens de emergência aos contatos privados dos usuários que tomarão conhecimento de qualquer incidente ou acidente ocorrido com eles.

Ficará para os futuros trabalhos, o aprimoramento do contato do mesmo aplicativo para o envio das mensagens de emergência através do meio público de comunicação direcionado não tão somente aos contatos privados do usuário, mas, também, as autoridades de emergência, o que acelerará o socorro e o resgate ao acidentado.

Ficará também para o futuro, a explanação do desenvolvimento do Módulo ECU veicular que acionará de forma automática o aplicativo de envio das mensagens, assim como, poderá também enviar de forma independente uma mensagem de emergência pelos canais *WiFi* oficiais.

REFERÊNCIAS

[1] ANTONIO MARQUES, MARCO. **Can Automotivo Sistema de Monitoramento - UNIFEI –** 2004. Disponível em http://saturnounifei.edu.br/bim/0030994pdf.

[2] CORREIA, SERGIO LUIS O B; CELESTINO, JOAQUIM; CHERKAOUI, OMAR. **Mobility-aware ant colony optimization routing for vehicular ad hoc networks In: Wireless Communications and Networking Conference (WCNC). IEEE [SI: sn], 2011 p 1125 –1130 ISSN 1525- 3511.** 2011.

[3] ECKERMANN, ERIK. **World History of the automobile - 1ª edição**. 2001

[4] FURNKRANZ, JOHANNES. **Round Robin Classification –** 03 de fevereiro de 2002. Disponível em http://wwwjmlr.org/papers/volume2/fuernkranz02a/fuernkranz02a.pdf. Acessado em 05 de julho de 2015.

[5] IEEE802.ORG. [2---]. Disponível em http://wwwieee802org/11/. Acessado em 11 de junho de 2015.

[6]. ISO. [2---], Disponível em http://www.iso.org/iso/home/searchhtm?qt=11898&published=on&active_tab= standards&sort_by=rel. Acessado em 14 de junho de 2015

[7] MORIMOTO, CARLOS EDUARDO - Redes, Guia Prático: ampliada e atualizada - 2011 - 2ª Edição

[8] MORIMOTO, CARLOS EDUARDO. 02 de agosto de 2011. Disponível em http://wwwhardwarecombr/guias/historia-informatica/eniac.html. Acessado em 14 de maio de 2015.

[9] MORENO, JOÃO BRUNELI. [2011]. Disponível em https://tecnoblognet/56910/eniac-primeiro-computador-do-mundo-completa-65-anos/. Acessado em 11 de junho de 2015.

[10] SANTOS, MMD. **Redes de comunicação automotiva – características, tecnologias e aplicações – São Paulo – 1ª edição**. 2010.

[11] SILVEIRA, THIAGO LOPES TRUGILLO DA. **Aplicação de Algoritmos de Escalonamento de Processos para Gerenciamento de Interseções em VANETS** – 6-8 de novembro de 2013. Disponível em http://www-usr.inf.ufsm.br/~thiago/public/Silveira,%20TLT%20and%20Pasin%20Marcia%20-%20ERRC2013.pdf. Acessado em 05 de julho de 2015.

[12] STITLIADIS, D & VARMA. **Efficient fair queueing algorithms for packet-switched networks –** 1998.

[13] TEXAS INTRUMENTS. **APLICATION REPORTS - SLOA101A -** August 2002 - Revised July 2008. Disponível em http://www.ti.com/lit/an/sloa101a/sloa101apdf. Acessado em 15 de junho de 2015.

[14] THE FUTURE OF THINGS. [2009]. Disponível em http://thefutureofthings.com/3898-the-future-of-wimax/. Acessado em 14 de junho de 2015.

[15] UZCATEGUI, R; ACOSTA,MARUM. **G Wave: a tutorial IEEE Communications Magazine, New York, v 47, n 5, p 126-133**. Maio de 2009.

[16] WIKPEDIA. **ALOHAnet -** 26 de dezembro de 2014. Disponível em http://ptwikipediaorg/wiki/ALOHAnet. Acessado em 11 de junho de 2015.

[17] WIKIPEDIA. **Vehicular ad hoc network –** 24 de janeiro de 2016. Disponível em http://www.digplanet.com/wiki/Vehicular_ad-hoc_network. Acessado em 10 de junho de 2015.

[18] ZHANG, H & KESHAV, S. **"Comparison of rate-based service disciplines" 'Proceedings of the conference on Communications architecture & proto-cols', ACM, 121 – 1991.**

[19] ZHANG, H & KNIGHTLY, E**. 'RCSP and stop-and-go: A comparison of two non-work-conserving disciplines for supporting multimedia communication',** ***Multimedia Systems*** **4(6), 346--356 – 1996.**

[20] WEIZER, MARK. **The Computer for the 21st Century –** [2---]. Disponível em https://www.ics.uci.edu/~corps/phaseii/Weiser-Computer21stCentury-SciAm.pdf. Acessado em 05 de janeiro de 2016.

APÊNDICE – Programação da Aplicação Crash Locator

- Arquivo Manifest.xml:

```
<?xml version="10" encoding="utf-8"?>
<manifest xmlns:android="http://schemasandroidcom/apk/res/android"
   package="brcommotbankcrashlocator" >

   <uses-permission android:name="androidpermissioninternet" />
   <uses-permission
android:name="androidpermissionACCESS_COARSE_LOCATION" />
   <uses-permission
android:name="androidpermissionACCESS_FINE_LOCATION" />
   <uses-permission android:name="androidpermissionINTERNET" />
   <uses-permission
android:name="androidpermissionACCESS_NETWORK_STATE" />
   <uses-permission
android:name="androidpermissionWRITE_EXTERNAL_STORAGE" />
   <uses-permission
android:name="comgoogleandroidprovidersgsfpermissionREAD_GSERVICES"
/>

   <application
      android:allowBackup="true"
      android:icon="@mipmap/crash"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
      <activity
         android:name="MainActivity"
         android:label="@string/app_name" >
         <intent-filter>
            <action android:name="androidintentactionMAIN" />

            <category android:name="androidintentcategoryLAUNCHER" />
         </intent-filter>
      </activity>
      <activity
         android:name="Configurar"
         android:label="@string/title_activity_configurar" >
      </activity>
      <activity
         android:name="Gps"
         android:label="@string/title_activity_gps" >
      </activity>

      <meta-data
         android:name="comgoogleandroidgmsversion"
         android:value="@integer/google_play_services_version" />
      <meta-data
         android:name="comgoogleandroidmapsv2API_KEY"
         android:value="@string/google_maps_key" />
```

```
    </application>

</manifest>
```

- Classe BancoController (faz conexão do banco de dados):

```
package brcommotbankcrashlocator;

import androidcontentContentValues;
import androidcontentContext;
import androiddatabaseCursor;
import androiddatabaseSQLException;
import androiddatabasesqliteSQLiteDatabase;
import androidutilLog;

import javautilArrayList;
import javautilList;

/**
 * Created by Alex on 14/01/16
 */
public class BancoController {
    private SQLiteDatabase db;
    private CriaBanco banco;

    public BancoController(Context context){
        banco = new CriaBanco(context); }


    public String insereContato(Contato contato){
        ContentValues valores;
        long resultado;
        db = bancogetWritableDatabase();
        valores = new ContentValues();
        valoresput(CriaBancoSMS1, contatogetSms1());
        valoresput(CriaBancoSMS2, contatogetSms2());
        valoresput(CriaBancoSMS3, contatogetSms3());
        valoresput(CriaBancoEMAIL1, contatogetEmail1());
        valoresput(CriaBancoEMAIL2, contatogetEmail2());
        valoresput(CriaBancoEMAIL3, contatogetEmail3());
        valoresput(CriaBancoWHATS1, contatogetWhats1());
        valoresput(CriaBancoWHATS2, contatogetWhats2());
        valoresput(CriaBancoWHATS3, contatogetWhats3());
        resultado = dbinsert(CriaBancoTBL_CONTATOS, null, valores);
        dbclose();
        if (resultado ==-1)
            return "Erro ao inserir contato";
        else
            return "Contatos inseridos com sucesso";
    }
```

```
    public boolean deletar(){
        boolean aux = true;
        try {
            db = bancogetWritableDatabase();
            String sql = "DELETE * FROM contatos";
            dbexecSQL(sql);
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            aux=false;
            Logd("Exception excluir", egetMessage()toString());
        }
        return aux;
    }

    public Contato carregaContatos(){
        //Open connection to read only
        SQLiteDatabase db = bancogetReadableDatabase();
        String selectQuery =  "SELECT  " +
                CriaBancoSMS1 + "," +
                CriaBancoSMS2 + "," +
                CriaBancoSMS3 + "," +
                CriaBancoEMAIL1 + "," +
                CriaBancoEMAIL2 + "," +
                CriaBancoEMAIL3 + "," +
                CriaBancoWHATS1 + "," +
                CriaBancoWHATS2 + "," +
                CriaBancoWHATS3 +
                " FROM " + CriaBancoTBL_CONTATOS;

        Cursor cursor = dbrawQuery(selectQuery, null);
        Contato contato = new Contato();

        if (cursormoveToFirst()) {
            do {

contatosetSms1(cursorgetString(cursorgetColumnIndex(CriaBancoSMS1)));

contatosetSms2(cursorgetString(cursorgetColumnIndex(CriaBancoSMS2)));

contatosetSms3(cursorgetString(cursorgetColumnIndex(CriaBancoSMS3)));

contatosetEmail1(cursorgetString(cursorgetColumnIndex(CriaBancoEMAIL1)));

contatosetEmail2(cursorgetString(cursorgetColumnIndex(CriaBancoEMAIL2)));

contatosetEmail3(cursorgetString(cursorgetColumnIndex(CriaBancoEMAIL3)));

contatosetWhats1(cursorgetString(cursorgetColumnIndex(CriaBancoWHATS1)));

contatosetWhats2(cursorgetString(cursorgetColumnIndex(CriaBancoWHATS2)));
```

```
contatosetWhats3(cursorgetString(cursorgetColumnIndex(CriaBancoWHATS3)));

            } while (cursormoveToNext());
        }

        cursorclose();
        dbclose();
        return contato;

    }

}
```

- Classe CriaBanco

```
package brcommotbankcrashlocator;

import androidcontentContext;
import androiddatabaseCursor;
import androiddatabaseMatrixCursor;
import androiddatabaseSQLException;
import androiddatabasesqliteSQLiteDatabase;
import androiddatabasesqliteSQLiteOpenHelper;
import androidutilLog;

import javautilArrayList;

/**
 * Created by Alex on 09/11/2015
 */
public class CriaBanco extends SQLiteOpenHelper {

    public static final String NOME_BANCO = "emergenciadb";
    public static final String TBL_CONTATOS = "contatos";
    public static final String ID = "_id";
    public static final String SMS1 = "sms1";
    public static final String SMS2 = "sms2";
    public static final String SMS3 = "sms3";
    public static final String EMAIL1 = "email1";
    public static final String EMAIL2 = "email2";
    public static final String EMAIL3 = "email3";
    public static final String WHATS1 = "whats1";
    public static final String WHATS2 = "whats2";
    public static final String WHATS3 = "whats3";
    private static final int VERSAO = 1;
    public static final String string1 = "CREATE TABLE "
            +TBL_CONTATOS+"("
            + ID + " integer primary key autoincrement, "
            + SMS1 + " text,"
```

```
            + SMS2 + " text,"
            + SMS3 + " text,"
            + EMAIL1 + " text,"
            + EMAIL2 + " text,"
            + EMAIL3 + " text,"
            + WHATS1 + " text,"
            + WHATS2 + " text,"
            + WHATS3 + " text)";

    public CriaBanco(Context context) {
        super(context, NOME_BANCO, null, 1);
    }
        @Override
    public void onCreate(SQLiteDatabase db) {
            dbexecSQL(string1);
        }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        dbexecSQL("DROP TABLE IF EXISTS contatos");

        onCreate(db);
    }

    public ArrayList<Cursor> getData(String Query){

//get writable database

        SQLiteDatabase sqlDB = thisgetWritableDatabase();

        String[] columns = new String[] { "mesage" };

//an array list of cursor to save two cursors one has results from the query

//other cursor stores error message if any errors are triggered

        ArrayList<Cursor> alc = new ArrayList<Cursor>(2);

        MatrixCursor Cursor2= new MatrixCursor(columns);

        alcadd(null);

        alcadd(null);


        try{

            String maxQuery = Query ;
```

```
//execute the query results will be save in Cursor c

        Cursor c = sqlDBrawQuery(maxQuery, null);


//add value to cursor2

        Cursor2addRow(new Object[] { "Success" });


        alcset(1,Cursor2);

        if (null != c && cgetCount() > 0) {



            alcset(0,c);

            cmoveToFirst();


            return alc ;
        }

        return alc;
    } catch(SQLException sqlEx){

        Logd("printing exception", sqlExgetMessage());


//if any exceptions are triggered save the error message to cursor an return the
arraylist

        Cursor2addRow(new Object[] { ""+sqlExgetMessage() });

        alcset(1,Cursor2);
        return alc;
    }

    catch(Exception ex){

        Logd("printing exception", exgetMessage());


//if any exceptions are triggered save the error message to cursor an return the
arraylist

        Cursor2addRow(new Object[] { ""+exgetMessage() });
        alcset(1,Cursor2);
```

```
            return alc;
        }


    }


}
```

- Classe configurar (cadastra os contatos):

```
package brcommotbankcrashlocator;

import androidgraphicsBitmap;
import androidgraphicsdrawableBitmapDrawable;
import androidsupportv7appAppCompatActivity;
import androidosBundle;
import androidviewMenu;
import androidviewMenuItem;
import androidviewView;
import androidwidgetSpinner;
import androidwidgetTextView;
import androidwidgetToast;

public class Configurar extends AppCompatActivity {
    private TextView txtSms1;
    private TextView txtSms2;
    private TextView txtSms3;
    private TextView txtEmail1;
    private TextView txtEmail2;
    private TextView txtEmail3;
    private TextView txtWhats1;
    private TextView txtWhats2;
    private TextView txtWhats3;
    private BancoController crud;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        superonCreate(savedInstanceState);
        setContentView(Rlayoutactivity_configurar);
        crud = new BancoController(getBaseContext());
        txtSms1 = (TextView) findViewById(Ridtxtsms1);
        txtSms2 = (TextView) findViewById(Ridtxtsms2);
        txtSms3 = (TextView) findViewById(Ridtxtsms3);
        txtWhats3 = (TextView) findViewById(Ridtxtwhats1);
        txtWhats2 = (TextView) findViewById(Ridtxtwhats2);
        txtWhats1 = (TextView) findViewById(Ridtxtwhats3);
        txtEmail3 = (TextView) findViewById(Ridtxtemail1);
        txtEmail2 = (TextView) findViewById(Ridtxtemail2);
        txtEmail1 = (TextView) findViewById(Ridtxtemail3);
```

```
        Carrega(crudcarregaContatos());

    }
    public void Carrega (Contato contato){
        txtSms1setText(contatogetSms1());
        txtSms2setText(contatogetSms2());
        txtSms3setText(contatogetSms3());
        txtEmail1setText(contatogetEmail1());
        txtEmail2setText(contatogetEmail2());
        txtEmail3setText(contatogetEmail3());
        txtWhats1setText(contatogetWhats1());
        txtWhats2setText(contatogetWhats2());
        txtWhats3setText(contatogetWhats3());
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present
        getMenuInflater()inflate(Rmenumenu_configurar, menu);
        return true;
    }

    public void Cadastrar(View v){
        crud deletar();
        Contato contato = new Contato();
        contatosms1 = txtSms1getText()toString();
        contatosms2 = txtSms2getText()toString();
        contatosms3 = txtSms3getText()toString();
        contatowhats1 = txtWhats1getText()toString();
        contatowhats2 = txtWhats2getText()toString();
        contatowhats3 = txtWhats3getText()toString();
        contatoemail1 = txtEmail1getText()toString();
        contatoemail2 = txtEmail2getText()toString();
        contatoemail3 = txtEmail3getText()toString();
        String resultado;
        resultado = crudinsereContato(contato);
        ToastmakeText(getApplicationContext(), resultado,
ToastLENGTH_LONG)show();
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifestxml
        int id = itemgetItemId();

        //noinspection SimplifiableIfStatement
        if (id == Ridaction_settings) {
            return true;
```

```
        }

        return superonOptionsItemSelected(item);
    }
}
```

- Classe Contato (informações dos contatos de emergência):

```
package brcommotbankcrashlocator;

/**
 * Created by Alex on 12/01/2016
 */
public class Contato {

    String sms1, sms2, sms3, email1,email2,email3,whats1,whats2,whats3;

    public Contato(){

    }

    public String getSms1() {
        return sms1;
    }

    public Contato(String sms1) {
        thissms1 = sms1;
    }

    public void setSms1(String sms1) {
        thissms1 = sms1;

    }

    public String getSms2() {
        return sms2;
    }

    public void setSms2(String sms2) {
        thissms2 = sms2;
    }

    public String getSms3() {
        return sms3;
    }

    public void setSms3(String sms3) {
        thissms3 = sms3;
    }
```

```
    public String getEmail1() {
        return email1;
    }

    public void setEmail1(String email1) {
        thisemail1 = email1;
    }

    public String getEmail2() {
        return email2;
    }

    public void setEmail2(String email2) {
        thisemail2 = email2;
    }

    public String getEmail3() {
        return email3;
    }

    public void setEmail3(String email3) {
        thisemail3 = email3;
    }

    public String getWhats1() {
        return whats1;
    }

    public void setWhats1(String whats1) {
        thiswhats1 = whats1;
    }

    public String getWhats2() {
        return whats2;
    }

    public void setWhats2(String whats2) {
        thiswhats2 = whats2;
    }

    public String getWhats3() {
        return whats3;
    }

    public void setWhats3(String whats3) {
        thiswhats3 = whats3;
    }

    public Contato(String sms1, String sms2, String sms3, String email1, String
email2, String email3, String whats1, String whats2, String whats3) {
```

```
        thissms1 = sms1;
        thissms2 = sms2;
        thissms3 = sms3;
        thisemail1 = email1;
        thisemail2 = email2;
        thisemail3 = email3;

        thiswhats1 = whats1;
        thiswhats2 = whats2;
        thiswhats3 = whats3;
    }
}
```

- Classe Gps (chama a classe que obtêm a localização do usuário):

```
package brcommotbankcrashlocator;

import androidcontentContext;
import androidlocationCriteria;
import androidlocationLocation;
import androidlocationLocationManager;
import androidsupportv7appAppCompatActivity;
import androidosBundle;
import androidviewMenu;
import androidviewMenuItem;
import androidwidgetTextView;
import androidwidgetToast;

public class Gps extends AppCompatActivity {
    private TextView txtLatitude;
    private TextView txtLongitude;
    private LocationManager locationManager;
    private String provider;
    GPSTracker gps;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        superonCreate(savedInstanceState);
        setContentView(Rlayoutactivity_gps);
        txtLatitude = (TextView) findViewById(RidtxtLatitude);
        txtLongitude = (TextView) findViewById(RidtxtLongitude);

        gps = new GPSTracker(Gpsthis);

        // check if GPS enabled
        if(gpscanGetLocation()){

            double latitude = gpsgetLatitude();
            double longitude = gpsgetLongitude();

            txtLatitudesetText(StringvalueOf(latitude));
```

```
            txtLongitudesetText(StringvalueOf(longitude));
        }else{
            // can't get location
            // GPS or Network is not enabled
            // Ask user to enable GPS/network in settings
            gpsshowSettingsAlert();
        }
    }


}
```

- Classe GpsTracker (obtêm a localização do usuário):

```
package brcommotbankcrashlocator;

import androidappAlertDialog;
import androidappService;
import androidcontentContext;
import androidcontentDialogInterface;
import androidcontentIntent;
import androidlocationLocation;
import androidlocationLocationListener;
import androidlocationLocationManager;
import androidosBundle;
import androidosIBinder;
import androidproviderSettings;
import androidutilLog;

/**
 * Created by Alex on 14/01/2016
 */
public class GPSTracker extends Service implements LocationListener {

    private final Context mContext;

    // flag for GPS status
    boolean isGPSEnabled = false;

    // flag for network status
    boolean isNetworkEnabled = false;

    // flag for GPS status
    boolean canGetLocation = false;

    Location location; // location
    double latitude; // latitude
    double longitude; // longitude

    // The minimum distance to change Updates in meters
```

```
    private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 10; // 10
meters

    // The minimum time between updates in milliseconds
    private static final long MIN_TIME_BW_UPDATES = 1000 * 60 * 1; // 1 minute

    // Declaring a Location Manager
    protected LocationManager locationManager;

    public GPSTracker(Context context) {
        mContext = context;
        getLocation();
    }


    public Location getLocation() {
        try {
            locationManager = (LocationManager)
mContextgetSystemService(LOCATION_SERVICE);

            // getting GPS status
            isGPSEnabled =
locationManagerisProviderEnabled(LocationManagerGPS_PROVIDER);

            // getting network status
            isNetworkEnabled =
locationManagerisProviderEnabled(LocationManagerNETWORK_PROVIDER);

            if (!isGPSEnabled && !isNetworkEnabled) {
                // no network provider is enabled
            } else {
                thiscanGetLocation = true;
                // First get location from Network Provider
                if (isNetworkEnabled) {

locationManagerrequestLocationUpdates(LocationManagerNETWORK_PROVIDER
,
                            MIN_TIME_BW_UPDATES,
                            MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
                    Logd("Network", "Network");
                    if (locationManager != null) {
                        location = locationManager

getLastKnownLocation(LocationManagerNETWORK_PROVIDER);
                        if (location != null) {
                            latitude = locationgetLatitude();
                            longitude = locationgetLongitude();
                        }
                    }
                }
```

```
            // if GPS Enabled get lat/long using GPS Services
            if (isGPSEnabled) {
                if (location == null) {
                    locationManagerrequestLocationUpdates(
                            LocationManagerGPS_PROVIDER,
                            MIN_TIME_BW_UPDATES,
                            MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
                    Logd("GPS Enabled", "GPS Enabled");
                    if (locationManager != null) {
                        location = locationManager
                                getLastKnownLocation(LocationManagerGPS_PROVIDER);
                        if (location != null) {
                            latitude = locationgetLatitude();
                            longitude = locationgetLongitude();
                        }
                    }
                }
            }
        }

    } catch (Exception e) {
        eprintStackTrace();
    }

    return location;
}

/**
 * Stop using GPS listener
 * Calling this function will stop using GPS in your app
 * */
public void stopUsingGPS(){
    if(locationManager != null){
        locationManagerremoveUpdates(GPSTrackerthis);
    }
}

/**
 * Function to get latitude
 * */
public double getLatitude(){
    if(location != null){
        latitude = locationgetLatitude();
    }

    // return latitude
    return latitude;
}

/**
```

```
     * Function to get longitude
     * */
    public double getLongitude(){
        if(location != null){
            longitude = locationgetLongitude();
        }

        // return longitude
        return longitude;
    }

    /**
     * Function to check GPS/wifi enabled
     * @return boolean
     * */
    public boolean canGetLocation() {
        return thiscanGetLocation;
    }

    /**
     * Function to show settings alert dialog
     * On pressing Settings button will lauch Settings Options
     * */
    public void showSettingsAlert(){
        AlertDialogBuilder alertDialog = new AlertDialogBuilder(mContext);

        // Setting Dialog Title
        alertDialogsetTitle("GPS is settings");

        // Setting Dialog Message
        alertDialogsetMessage("GPS is not enabled Do you want to go to settings
menu?");

        // On pressing Settings button
        alertDialogsetPositiveButton("Settings", new DialogInterfaceOnClickListener()
{
            public void onClick(DialogInterface dialog,int which) {
                Intent intent = new
Intent(SettingsACTION_LOCATION_SOURCE_SETTINGS);
                mContextstartActivity(intent);
            }
        });

        // on pressing cancel button
        alertDialogsetNegativeButton("Cancel", new DialogInterfaceOnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                dialogcancel();
            }
        });
```

```
        // Showing Alert Message
        alertDialogshow();
    }

    @Override
    public void onLocationChanged(Location location) {
    }

    @Override
    public void onProviderDisabled(String provider) {
    }

    @Override
    public void onProviderEnabled(String provider) {
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
    }

    @Override
    public IBinder onBind(Intent arg0) {
        return null;
    }

}
```

- Classe Principal:

```
package brcommotbankcrashlocator;

import androidcontentIntent;
import androidsupportv7appAppCompatActivity;
import androidosBundle;
import androidviewMenu;
import androidviewMenuItem;
import androidviewView;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        superonCreate(savedInstanceState);
        setContentView(Rlayoutactivity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present
```

```
        getMenuInflater()inflate(Rmenumenu_main, menu);
        return true;
    }

    public void Configurar (View v){
        Intent i = new Intent(MainActivitythis, Configurarclass);
        startActivity(i);
    }

    public void Alertar (View v){
        Intent i = new Intent(MainActivitythis, Gpsclass);
        startActivity(i);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifestxml
        int id = itemgetItemId();

        //noinspection SimplifiableIfStatement
        if (id == Ridaction_settings) {
            return true;
        }

        return superonOptionsItemSelected(item);
    }
}
```